SPECIFICATION AMENDMENTS:

Please amend the paragraph beginning at page 2, line 7 as follows:

"The ~~Java™~~ JAVA programming language developed by Sun Microsystems, Inc. includes a StreamTokenizer class. This tokenizer class only accepts single character delimiter-tokens that it refers to as ordinary characters. The StreamTokenizer class does not accept multi-character delimiter-tokens."

Please amend the paragraph beginning at page 2, line 11 as follows:

"The ability to recognize string delimiters and string delimiter-tokens is a capability that is important for processing some contemporary programming languages, such as ~~Java~~ JAVA programming language and HTML (hypertext mark up language). Accordingly, there is a need for an improved lexical analyzer and method that recognize string delimiters and delimiter-tokens."

Please amend the paragraph beginning at page 3, line 9 as follows:

"The present invention results in a lexical analyzer that is capable of tokenizing contemporary programming languages, such as ~~Java™~~ JAVA programming language, that include multi-character delimiters and delimiter-tokens."

Please amend the paragraph beginning at page 3, line 18 as follows:

"FIG. 3 is a decision table illustrating the actions taken ~~be~~ by the lexical analyzer during its operation."

Please amend the paragraph beginning at page 3, line 18 as follows:

"As another example of string delimiter-tokens, the ~~Java~~ JAVA Server Page (JSP) and HTML (hypertext markup language) comment strings are tokens and also act as delimiters. For instance, the string "<--" (excluding the double quotes) is a begin comment string in JSP. The string "-->" (excluding the double quotes) is an end comment string in JSP. Any character(s) occurring between a delimiter and one of the comment strings are returned together as a token. The comment string can be subsequently returned by the lexical analyzer 12 as a token. The ability to define and identify multi-character special delimiter-tokens is the advantage of the lexical analyzer 12. "

Please amend the paragraph beginning at page 6, line 1 as follows:

"FIG. 2 is a flowchart 30 illustrating a method of operating the lexical analyzer 12 of FIG. 1, in accordance with the present invention. In general terms, the lexical analyzer 12 performs a looping operation until a token has been determined in an input stream, and returns the token to the calling program. A token is determined when either a delimiter or a delimiter-token has been determined. If a delimiter-token has been determined, then in a subsequent call to the lexical analyzer 12, the delimiter-token is returned as token. Special handling is required when multi-character delimiters are used."

Please amend the paragraph beginning at page 7, line 1 as follows:

"If the character variable is not a delimiter, the method proceeds to step 44, where a check is made to determine whether the character variable is a delimiter-token. This is accomplished by the detector 24 comparing the character variable to the delimiter-token table 22. If character is not a delimiter-token, the token variable is set to the character variable (step 52), and the lexical analyzer 12 gets the next character from the input stream ~~54~~ 56."

Please amend the paragraph beginning at page 7, line 15 as follows:

"However, if the character variable is not a delimiter, a check is made to determine whether the character variable represents a delimiter-token (step ~~42~~ 41). If so, the lexical analyzer 12 returns the token variable (step 50). If not, the value of the character variable is appended to the token variable (step ~~44~~ 43)."

Please amend the paragraph beginning at page 8, line 15 as follows:

"The lexical analyzer and method described herein can be used to identify deprecated statements in software code that is being migrated to newer versions of a particular programming language, such as ~~Java™~~ JAVA programming language. Deprecated statements are software constructs or language that are no longer supported by later versions of a language. According to one embodiment of the invention, symbols representing deprecated statements can be entered into the delimiter-token tables 18, 22 to be specifically identified by the lexical analyzer 12. Alternatively, the lexical analyzer 12 can return tokens representing deprecated statements, the tokens being identified as such by the application software 14."

Please amend the Abstract as attached hereto.